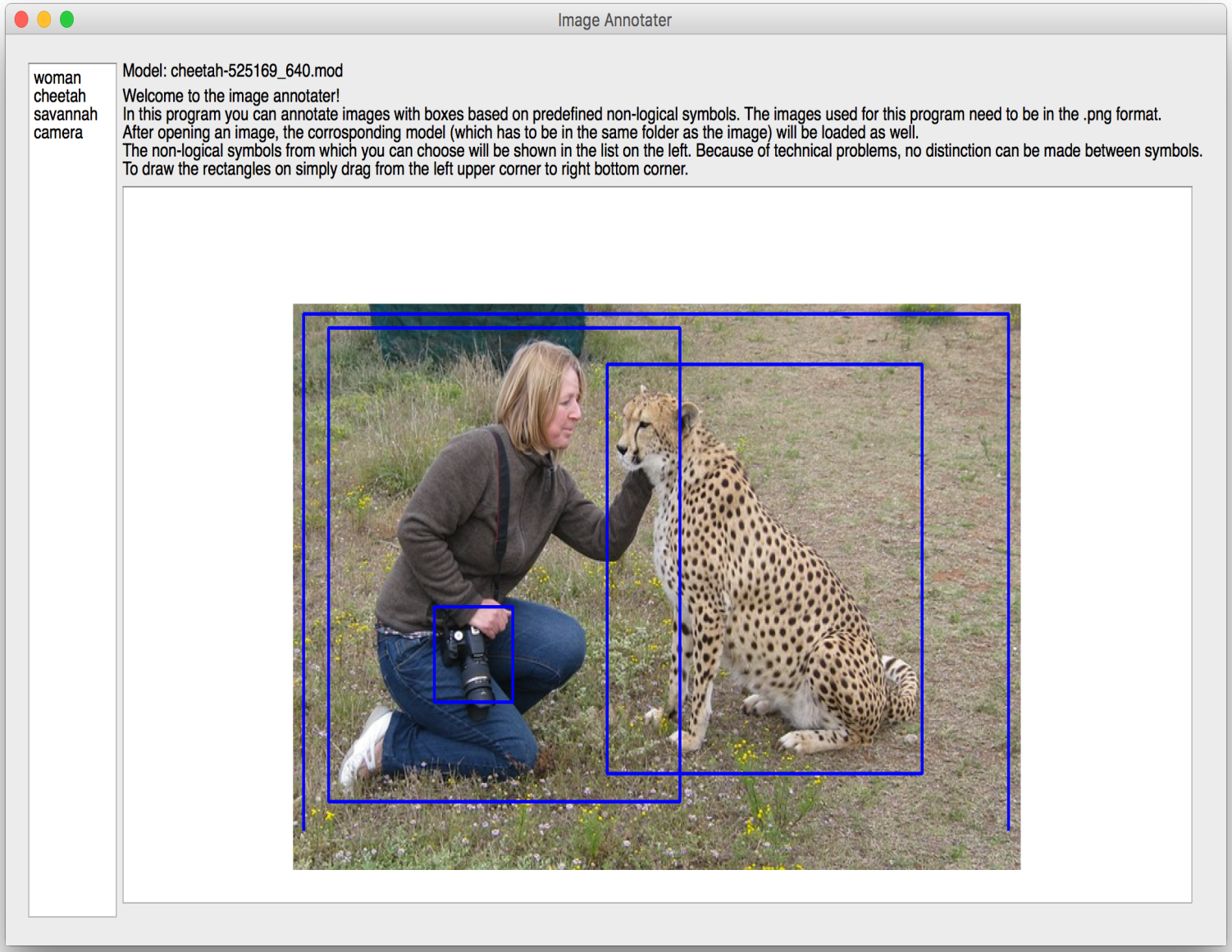
Computational Semantics

Final Project: Image model authoring tool



Mart Busger op Vollenbroek
S2174634

Introduction

Problem description

For the course Computational Semantics a final project had to be made. We could choose between four different final projects and I have chosen to make the fourth final project: an image model authoring tool. According to the assignment, the system had several requirements:
- A simple graphical interface to add bounded boxes to the image models (“extended Tarski models");
- Input: image model; output: extended models, with bounded boxes;
- Making suggestions for spatial relations;
- EXTRA MILE: consistency checking of spatial relations and bounded boxes.

The graphical interface will be written in python, making use of the PyQt4 library for the graphical elements. The task at hand is to develop a system that takes in an image and a corresponding model and output the image enriched with boxes (rectangles) and the model enriched with the geo data.

Relevant literature

Models are simplifications from reality or, a little bit more realistic, approximations of reality. This is because computers can not show a hundred percent true representation of reality. In order to get such an approximation of reality, several tools and some knowledge is needed. First of all, first order logic is needed. “First-order logic is the formalism to represent the meaning of natural language sentences and to carry out various inference tasks”, according to Blackburn & Bos (2005). First order logic consists of syntax and semantics, which in turn are described using vocabularies, first order languages and first order models. Vocabularies are the basis for first order languages and first order models, because vocabularies link those two to each other. A vocabulary consists of all the relevant subjects in the representation of reality, so it basically explains what the model is about. Vocabularies also explain how the relevant subjects of reality are used.
First order languages consist of logical symbols and non-logical symbols. Examples of logical symbols are Boolean values such “True”, “False”, “And”, “Or" and “Not”, but also implication and universal and existential quantifiers. The relevant subjects in the reality to be described are called the non-logical symbols. These non-logical symbols are different for every model, because no model is the same. However, models can contain the same non-logical symbols. To avoid having to deal with the same, ambiguous non-logical symbols across models, a lexicon of some kind is needed. WordNet is such a lexical resource and it provides a rich dictionary with concepts and their meanings (Miller, 1995). WordNet has a hierarchical structure, which means that all concepts have higher level concepts which finally all lead to the same top-level concept. An example of this is an actor. An actor is a hyponym of a performer, which is a hyponym of an entertainer, which in turn is a hyponym of a person. Going up some more levels in the tree leads to the top-level synset which is an "entity".
Extended models with bounded boxes are also called “extended Tarski models”, after Alfred Tarski. As Raatikainen (2008) mentions, Tarski’s assumption is as follows: “if the syntax of the object language is specified exactly enough, and that the metatheory has a certain amount of set theoretic power, one can explicitly define truth in the object language”. In this project, that means that if concepts are specified as detailed as possible, one can get a close approximation from reality.
For evaluating the system, the 10 heuristics for User Interface Design by Nielsen (1995) are used. These are not specific guidelines, but are general rules of thumb for evaluating interaction design. This summarizing information has been found on the Nielson Norman Group website (https://www.nngroup.com/articles/ten-usability-heuristics/) and is as follows:

Visibility of system status
The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

Match between system and the real world
The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

User control and freedom
Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

Consistency and standards
Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

Error prevention
Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

Recognition rather than recall
Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

Flexibility and efficiency of use
Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

Aesthetic and minimalist design
Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

Help users recognize, diagnose, and recover from errors
Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

Help and documentation
Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

Method

Data

Using the information gained from the literature and the rest of the course, a system was made to try and create extended Tarski models. The input for the system is collection of the images and the models which were made by students during the course. The images were provided at http://

www.let.rug.nl/bos/comsem/newimages/. The models can also be found here, but these were also provided in a tar file on nestor.
The models used for the final project have been made in the weeks leading up the final project. All students got assigned some of the images and were tasked to provide models for them in a predefined format. All models had to be defined in first order logic and in Prolog format, which made it easier for them to be checked for consistency by model checkers.
All models have a domain which contains all the entities in the specific model and assignment functions that map all variables to an entity in the domain. An example of such a model can be found in Figure 1. Figure 2 is the corresponding image to the model.

```
model([d1,d2,d3,d4],
      [f(1,n_pheasant_1,[d1,d2]),
       f(1,n_grass_1,[d3]),
       f(1,n_hedgehog_1,[d4]),
       f(2,s_touches,[(d1,d3),(d3,d1),(d2,d3),(d3,d2),(d3,d4),(d4,d4)]),
       f(2,s_supports,[(d3,d1),(d3,d2),(d3,d4)]),
       f(2,s_near,[(d1,d3),(d3,d1),(d2,d3),(d3,d2),(d3,d4),(d4,d4)])]).
```

Figure 1: Model example animals-110721_640

Figure 2: Image example animals-110721_640

In the example model, there exist four entities in the domain (e.g. d1, d2, d3, d4). These four entities have all been linked to a WordNet concept. As seen in the model, two entities can be assigned to the same WordNet concept. In this model, no WordNet adjectives have been used. Those would normally be shown in model with an ‘a' instead of the ’n’ for noun concepts.
All models also contain spatial information denoted by the 's'. The spatial relations used in the models are ‘touches’, ‘near’, ‘supports’, ‘part_of’ and ‘occludes’. From these spatial relations, several axioms can be formulated which describe the properties of those spatial relations. Examples of these axioms are ‘every concept x that supports another concept y also touches that other concept y’ and ‘every concept x that is near another concept y does not touch concept y’.

Using those axioms, extended Tarski models can be checked for the consistency of the bounded boxes.

System

The system itself should be able to read in the images and models discussed above and enrich them with bounded boxes and their coordinates respectively. For reading in the images and creating a graphical user interface, the PyQt4 library is used. When the system is started, the user is immediately prompted to select an image in the .png format. This is also the first difficulty, because the images supplied by the image corpus are all in the .jpg format. For testing purposes, some of the images were therefore converted to the right format so that they could be displayed in the GUI.
The GUI mainly consists of three parts: an instruction text explaining the system's possibilities, the list with the model's non-logical symbols and the model's corresponding image. At first the list with non-logical symbols was a QListWidget, but using that seemed to draw random bounded boxes on the images after drawing the first box. Because of this error, the QListWidget was replaced by a simple read-only QTextEdit and thereby any feature trying to combine a non-logical symbol to a bounded box was removed. Because of the errors seemingly created by QListWidget and subsequent removal of it from the system, the only task the system can perform is drawing bounded boxes on images and giving feedback on the position of the upper left corner and bottom right corner. An example of how that looks using the above mentioned model and image can be found in figure 3.

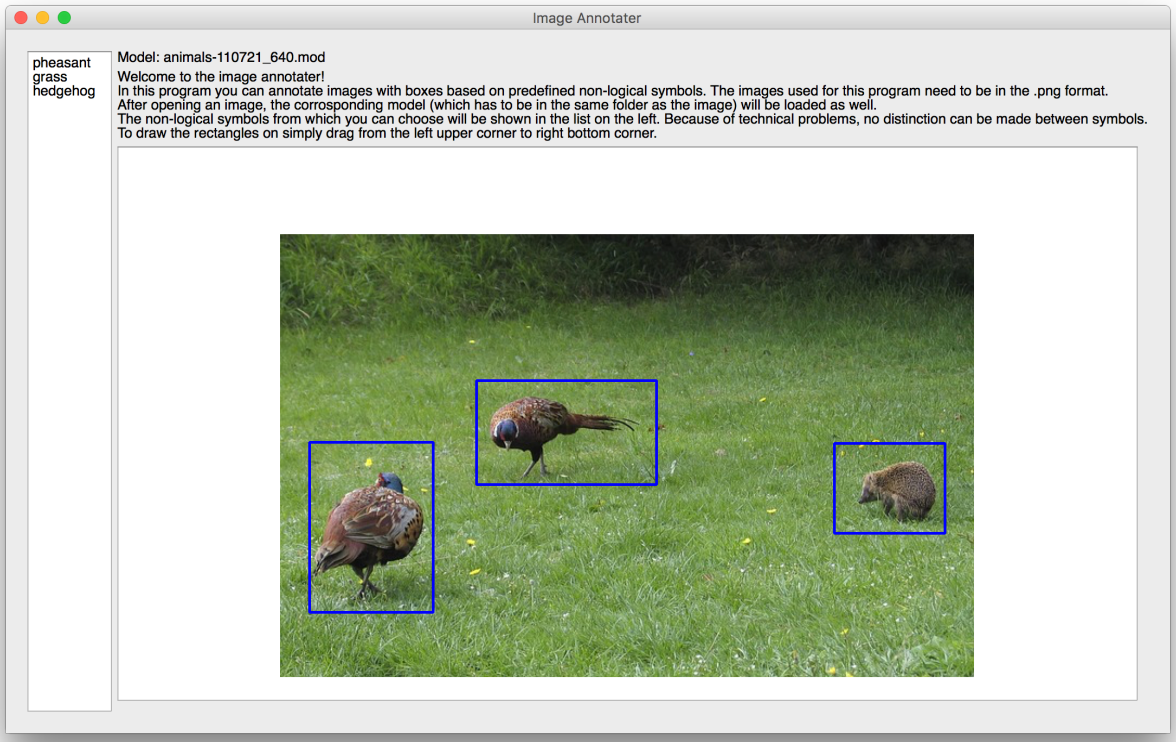


Figure 3: Final system output example

The feedback the user gets for every drawn rectangles is in the format:
Top left corner at X:181.0 Y:165.0
Bottom right corner at X:347.0 Y:261.0

Evaluation

Design evaluation

As mentioned before, the 10 Usability Heuristics for User Interface Design by Nielsen (1995) are used to evaluate the system's usability.

Visibility of system status
The system gives the user feedback on when the user has started drawing a bounded box and when the user has finished drawing a bounded box by informing the user on the coordinates from the bounded boxes.

Match between system and the real world
The instruction text within the system has been written using phrases and concepts which should be familiar to user, as the user also follows or teaches the course Computational Semantics.

User control and freedom
There is no elaborate user control or freedom within the program, because in essence it is a very simple and straightforward program. There are als no possibilities to undo or redo the drawing of the bounded boxes. This means that there is basically no user control or freedom at all.

Consistency and standards
Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

Error prevention
Because the system seemed to give errors when using the QListWidget for displaying the non-logical symbols, that widget has been removed from the system. Another error occurred while starting the system and loading the image, which resulted in very unclear feedback. By implementing a try-except in the code, this error has been removed.

Recognition rather than recall
Because the system is so straightforward, there are no different parts and all the information is visible at all times. The instruction text is always visible so if a user does not know how to proceed, that information can always be consulted.

Flexibility and efficiency of use
Again because of the simpleness of the system, there is no flexibility. This does mean however, that the system is efficient in what is does.

Aesthetic and minimalist design
The information given in the instructions is all relevant and can be needed at all times. The design in itself is also very minimalistic, which makes the system easy to use.

Help users recognize, diagnose, and recover from errors
Up until now, no further error messages have been noted to occur so the user is not given any help to recognize, diagnose or discover errors. However if a new error is encountered, the error message would be in code, so it might be unclear to user how to proceed.

Help and documentation
No documentation has been provided to use the system. There is only one instruction text within the program itself and that text explains all the possibilities.

Task evaluation

Unfortunately, no way has been found to save the images with their bounded boxed, nor has a way been found to enrich the models with this geographical information in this way. The model does not receive the information, and even if it would, there is no connection with the non-logical symbols from the domain and so the only information added to the model would be that there exist bounded boxed at certain coordinates.
There are also no suggestions given about spatial relations and the consistency of the spatial relations and bounded boxes isn't checked at all. This leads to the conclusion that the system performs badly given the task which it had to complete.

Bibliography

Blackburn, P., & Bos, J. (2005). Representation and inference for natural language. *A first course in computational semantics. CSLI*.

Miller, G. A. (1995). WordNet: a lexical database for English. *Communications of the ACM*, *38*(11), 39-41.

Nielsen, J. (1995). 10 Usability Heuristics for User Interface Evaluation. Checked 30-1-2016 at <<https://www.nngroup.com/articles/ten-usability-heuristics/>>

Raatikainen, P. (2008). Truth, correspondence, models, and Tarski. *Approaching truth: Essays in honour of Ilkka Niiniluoto*, 107-119.